日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年　８月１９日

出願番号
Application Number: 特願２００２－２３８３９９

[ST.10/C]: [ＪＰ２００２－２３８３９９]

出願人
Applicant(s): 富士通株式会社

２００３年　１月　７日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００２－３１０３５６５

Attorney Docket No. 1448.1040

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Hideo MIYAKE, et al.

Application No.: Group Art Unit:

Filed: June 25, 2003 Examiner:

For: METHOD OF AND APPARATUS FOR CREATING LOAD MODULE AND COMPUTER PRODUCT

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). 2002-187230 and 2002-238399

Filed: June 27, 2002 and August 19, 2002

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: June 25, 2003 By: _____

H. J. Staas
Registration No. 22,010

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

【書類名】　特許願

【整理番号】　0240872

【提出日】　平成14年 8月19日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 15/16

【発明の名称】　ロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置

【請求項の数】　9

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　三宅　英雄

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　上方　輝彦

【発明者】

【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

【氏名】　畔上　謙吾

【特許出願人】

【識別番号】　000005223

【氏名又は名称】　富士通株式会社

【代理人】

【識別番号】　100104190

【弁理士】

【氏名又は名称】　酒井　昭徳

【手数料の表示】

【予納台帳番号】　041759

【納付金額】　　21,000円

【提出物件の目録】

【物件名】　　明細書　1

【物件名】　　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9906241

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　ロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置

【特許請求の範囲】

【請求項１】　複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第１のメモリ空間構築工程と、

前記第１のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第１のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第１のメモリ空間内アドレス解決工程と、

第２のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第２のメモリ空間構築工程と、

前記第２のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第２のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第２のメモリ空間内アドレス解決工程と、

前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

を含んだことを特徴とするロードモジュール生成方法。

【請求項２】　前記メモリ空間間アドレス解決工程では、前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第２のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項１に記載のロードモジュール生成方法。

【請求項３】　前記第１のメモリ空間構築工程では、前記第１のプロセッサからのみ参照可能なメモリ領域と、前記第１のプロセッサおよび前記第２のプロ

セッサから参照可能なメモリ領域とを構築することを特徴とする請求項１または請求項２に記載のロードモジュール生成方法。

【請求項４】　複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第１のメモリ空間構築工程と、

前記第１のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第１のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第１のメモリ空間内アドレス解決工程と、

第２のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第２のメモリ空間構築工程と、

前記第２のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第２のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第２のメモリ空間内アドレス解決工程と、

前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【請求項５】　前記メモリ空間間アドレス解決工程では、前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第２のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項４に記載のロードモジュール生成プログラム。

【請求項６】　前記第１のメモリ空間構築工程では、前記第１のプロセッサからのみ参照可能なメモリ領域と、前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域とを構築することを特徴とする請求項４または

請求項５に記載のロードモジュール生成プログラム。

【請求項７】　複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において、

第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第１のメモリ空間構築手段と、

前記第１のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第１のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第１のメモリ空間内アドレス解決手段と、

第２のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第２のメモリ空間構築手段と、

前記第２のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第２のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第２のメモリ空間内アドレス解決手段と、

前記第１のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決手段と、

を備えたことを特徴とするロードモジュール生成装置。

【請求項８】　前記メモリ空間間アドレス解決手段は、前記第１のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスの、前記第２のメモリ空間構築手段により構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項７に記載のロードモジュール生成装置。

【請求項９】　前記第１のメモリ空間構築手段は、前記第１のプロセッサからのみ参照可能なメモリ領域と、前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域とを構築することを特徴とする請求項７または請求項８に記載のロードモジュール生成装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置に関する。

【０００２】

【従来の技術】

近年の計算機システムでは、複数のプロセッサを搭載することでシステムの処理能力を向上させるべく、「分散メモリ型マルチプロセッサ方式（Ｄｉｓｔｒｉｂｕｔｅｄ－Ｍｅｍｏｒｙ　Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒｓ）」が採用されることがある。

【０００３】

図１は、分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。図示するように、プロセッサ（ＰＲＯＣＥＳＳＯＲ））１０１とメモリ（ＭＥＭＯＲＹ）１０２とから構成されるＰＥ（Ｐｒｏｃｅｓｓｉｎｇ　Ｅｌｅｍｅｎｔ：プロセッサ要素）１００がｎ個、相互接続網（ＩＮＴＥＲＣＯＮＮＥＣＴＩＯＮ　ＮＥＴＷＯＲＫ）１０３により接続されている。

【０００４】

また、図２は上記システムにおけるメモリ空間の定義例を模式的に示す説明図である。図示するように個々のプロセッサ１０１は、同じＰＥ１００内のメモリ１０２だけを読み書きすることができる。

【０００５】

そしてこのようなシステムにおいては、ＭＰＩ（Ｍｅｓｓａｇｅ－Ｐａｓｓｉｎｇ　Ｉｎｔｅｒｆａｃｅ）などのプロセッサ間通信機構を用いることで、ＳＰＭＤ（Ｓｉｎｇｌｅ－Ｐｒｏｇｒａｍ，Ｍｕｌｔｉｐｌｅ－Ｄａｔａ）プログラミングにもとづくプログラムが実行されることが多い。

【０００６】

図３は上記プログラムの一例を示す説明図である。図示するプログラムはｎ個

のメモリ１０２にそれぞれ格納され、ｎ個のプロセッサ１０１によりそれぞれ実行される。プログラムは同一でも、ＰＥ１００のＩＤ（番号）により処理が分岐するので、ｎ個のＰＥ１００による並列処理が実現される。

【０００７】

たとえば図示するプログラムでは、「ｍｙ＿ｒａｎｋ」が上記ＩＤを示す変数であり、ｍｙ＿ｒａｎｋ＝０以外のＰＥではｉｆ以下の処理が、ｍｙ＿ｒａｎｋ＝０のＰＥではｅｌｓｅ以下の処理が、それぞれ実行されることになる。

【０００８】

また、図４は上記プログラムのロードモジュールの生成手順を示すフローチャートである。まずコンパイラにより、プログラムのソース記述をアセンブリ記述に変換し（ステップＳ４０１～Ｓ４０３）、次にアセンブラにより、上記アセンブリ記述からオブジェクトを生成する（ステップＳ４０４～Ｓ４０６）。そして、上記により生成された複数のオブジェクトをリンカにより結合することで（ステップＳ４０７～Ｓ４１０）、上記プログラムのロードモジュールを生成する。

【０００９】

ところで、図１のような分散メモリ型マルチプロセッサ方式にもとづくシステムは、従来は半導体集積技術の限界から、複数のチップ（および複数のボード）により構成されてきた。しかしながら、近年の半導体集積技術の向上により、複数のＰＥを一つのチップに収めることが可能となっている。

【００１０】

この場合、相互接続網を介したＰＥ間のデータの受け渡しはパケット伝送方式ではなく、共有メモリにデータを直接ストア／共有メモリからデータを直接ロードすることで、より高速におこなうことができる。このように、複数のプロセッサから読み書きされる共有メモリを設ける方式を、「分散共有メモリ型マルチプロセッサ方式」と呼ぶ。

【００１１】

図５は、分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。図１に示した分散メモリ型マルチプロセッサ方式との差異は、メモリ５０２に、他のＰＥ内のプロセッサからも読み書きできるＳ

Ｍ（Ｓｈａｒｅｄ　Ｍｅｍｏｒｙ：共有メモリ）と、同一のＰＥ内のプロセッサからしか読み書きできないＬＭ（Ｌｏｃａｌ　Ｍｅｍｏｒｙ：固有メモリ）との２種類ある点である。

【００１２】

また、図６は上記システムにおけるメモリ空間の定義例を示す説明図である。図中、たとえば１番のＰＥ（ＰＥ＃１）のＳＭは、０番のＰＥ（ＰＥ＃０）のメモリ空間および１番のＰＥ（ＰＥ＃１）のメモリ空間に重複して割り当てられている。

【００１３】

仮に、ＰＥ＃１のＳＭがＰＥ＃０のメモリ空間では０ｘ３０００以下、ＰＥ＃１のメモリ空間では０ｘ２０００以下のアドレスに割り当てられていたとすると、たとえばＰＥ＃０が０ｘ２０００にデータを書き込み、ＰＥ＃１が０ｘ３０００からデータを読み出すことで、ＰＥ＃０とＰＥ＃１との間で上記データを授受できたことになる。

【００１４】

なお、図示する例ではＰＥ＃０のみが、他のすべてのＰＥのＳＭを参照・変更することができる。一方ＰＥ＃１～＃ｎの各メモリ空間には、物理的に他のＰＥに属するメモリが割り当てられていないので、これらのＰＥは同一ＰＥ内のＬＭおよびＳＭを参照・変更するのみである。

【００１５】

そして、このような分散共有メモリ型マルチプロセッサ方式の計算機システムでも、分散メモリ型マルチプロセッサ方式と同様、図３に示したようなＳＰＭＤプログラミングにもとづくプログラムを実行することは可能である。

【００１６】

【発明が解決しようとする課題】

しかしながら、分散メモリ型マルチプロセッサ方式にせよ、分散共有メモリ型マルチプロセッサ方式にせよ、個々のプロセッサが実行するのはプログラムの一部（以下では「部分プログラム」という）であるにもかかわらず、各ＰＥにはプログラムの全体が配分されるので、それだけの容量のメモリを用意しなければな

らず、コストがかさんでしまうという問題があった。

【００１７】

この問題は、少なくとも分散メモリ型マルチプロセッサ方式のシステムについては、プログラムをＳＰＭＤでなくＭＰＭＤ（Ｍｕｌｔｉｐｌｅ－Ｐｒｏｇｒａｍ，Ｍｕｌｔｉｐｌｅ－Ｄａｔａ）プログラミングにもとづいて作成することで回避可能である。

【００１８】

ＭＰＭＤにもとづくプログラミングでは、ＳＰＭＤのように各ＰＥにより実行される部分プログラムをすべて結合したようなプログラムでなく、端的にそれぞれのＰＥ向けのプログラムを作成する。図７はＰＥ＃０、図８はＰＥ＃１～＃ｎ向けのプログラムの一例をそれぞれ示す説明図である。図示するように各ＰＥ用のプログラムには、他ＰＥ用の部分プログラムが含まれないので、その分メモリの容量を小さくすることができる。なお、これらのプログラムのロードモジュールは、図４に示した手順により生成される。

【００１９】

一方、分散共有メモリ型マルチプロセッサ方式では、上述のようにある場所に格納された同一のデータを複数のＰＥが参照・変更するが、メモリ空間上のそのアドレスはＰＥごとに異なっている。したがって、各ＰＥ向けのプログラムについてリンカでアドレスを解決する際には、対象物が同一でもＰＥごとに異なるアドレスに変換しなければならないが、従来技術のリンカにはこうした機能がなかった。

【００２０】

そのため分散共有メモリ型マルチプロセッサ方式の計算機システムで動作するプログラムは、ＳＰＭＤプログラミングによってしか作成することができず、実行時にはスキップされる部分プログラムが各ＰＥに多数配分され、無駄な部分プログラムを保持するために多くのメモリが必要になってしまうという問題があった。

【００２１】

この発明は上記従来技術による問題を解決するため、分散共有メモリ型マルチ

プロセッサ方式を採用する計算機システムにおいても、より少ないメモリ容量で動作するプログラムのロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置を提供することを目的とする。

【００２２】

【課題を解決するための手段】

上述した課題を解決し、目的を達成するため、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、第１のプロセッサにより実行されるプログラムおよび第２のプロセッサにより実行されるプログラムについて、それぞれオブジェクトを結合して所定のメモリ領域を構築のうえ各シンボルのアドレスを算出するとともに、第１のプロセッサにより実行されるプログラム中アドレスが解決されなかったシンボルのアドレスを、第２のプロセッサにより実行されるプログラム中での当該シンボルのアドレスにもとづいて算出することを特徴とする。

【００２３】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、第１のプロセッサにより実行されるプログラム中アドレスが解決されなかったシンボルのアドレスを、第２のプロセッサにより実行されるプログラム中での当該シンボルのアドレスの、当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする。

【００２４】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する際、第１のプロセッサからのみ参照可能なメモリ領域と、第１のプロセッサおよび第２のプロセッサから参照可能なメモリ領域とを構築することを特徴とする。

【００２５】

これらの発明によって、ＭＰＭＤプログラミングにもとづいて作成された各プロセッサ用のソースプログラムから、分散共有メモリ型マルチプロセッサ方式を採用する計算機でも実行可能なそのロードモジュールを生成することができる。逆にいえば、分散共有メモリ型マルチプロセッサ方式の計算機についても、ＭＰＭＤプログラミングによってプログラムを作成することができる。

【００２６】

【発明の実施の形態】

以下に添付図面を参照して、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置の好適な実施の形態を詳細に説明するが、その前に本発明の基本方針を簡単に説明する。

【００２７】

（基本方針）

図９および図１０は、分散共有メモリ型マルチプロセッサ方式の計算機システムを前提とする、ＭＰＭＤプログラミングにもとづくプログラムの一例を示す説明図である。図９はＰＥ＃０用、図１０はＰＥ＃１用であり、ＰＥ＃０からＰＥ＃１に必要なデータを渡して、所定の処理を依頼した後、その結果を受け取るためのプログラムである。

【００２８】

すなわちまずＰＥ＃０において、変数ｉｎｐｕｔを読み出して、その値を変数ｉｎに書き込み（図９Ｔｈ０－１）、次にＰＥ＃１の関数Ｔｈ１の実行を指示する（図９Ｔｈ０－２）。これを受けたＰＥ＃１では、Ｔｈ１の中で変数ｉｎを入力として関数ｆ１を呼び出し、その実行結果を変数ｏｕｔに書き込む（図１０Ｔｈ１－１）。その後、ＰＥ＃０は変数ｏｕｔを読み出し、その値を変数ｏｕｔｐｕｔに書き込む（図９Ｔｈ０－３）。

【００２９】

なお、実際のプログラムではＰＥ＃１に処理を依頼した後（すなわちＴｈ０－２の後）、ＰＥ＃０はＰＥ＃１とは無関係な別の処理に移行するが、ここでは簡略化してＰＥ＃０－ＰＥ＃１間の連携部分のみを示している。

【００３０】

上述のように、従来技術の言語処理系ではこれらのソースプログラムについて、実際に実行可能なロードモジュールを生成することができない。たとえば図９に示したＰＥ＃０用のプログラムで、ｅｘｔｅｒｎ宣言されている変数ｉｎおよびｏｕｔは、図１０に示したＰＥ＃１用のプログラムで定義されているため、ＰＥ＃０用のプログラムをリンクした時点ではアドレスは不定である。

【００３１】

ＰＥ＃１用のプログラムをリンクすると上記変数のアドレスは確定するが、判明するのはあくまでＰＥ＃１のメモリ空間上でのアドレスであり、当該アドレスで指し示される物理的な記憶領域の、ＰＥ＃０のメモリ空間上におけるアドレスは依然不明である。

【００３２】

そこで、以下に説明する実施の形態のような計算式を用いて、ＰＥ＃１のメモリ空間上でのアドレスから、ＰＥ＃０のメモリ空間上でのアドレスを割り出してやることで、ＰＥ＃０用のプログラムで未解決シンボルとして残った、変数ｉｎおよびｏｕｔのアドレスを解決する。

【００３３】

図１１は、図９および図１０に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図である。同一の変数ｉｎ／ｏｕｔが、ＰＥ＃０用のプログラムでは０ｘ３０００／０ｘ３００４に、ＰＥ＃１用のプログラムでは０ｘ２０００／０ｘ２００４に、それぞれ置換されていることが分かる。

【００３４】

なお、図中「ｔｅｘｔ　ａｒｅａ」とはプログラムの命令列を保持する領域であり、「ｄａｔａ　ａｒｅａ」とはプログラムから読み書きされるデータのうち、非共有データすなわち当該プログラムを実行するＰＥ以外の他のＰＥからは参照・変更されることのないデータを保持する領域である。これらの領域は物理的には各ＰＥのＬＭに配置され、他のＰＥからの参照や変更はできない。

【００３５】

また、図中「ｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ　＃ｋ」（０≦ｋ≦ｎ）は、

いずれも共有データを保持する領域である。これらの領域は物理的には、それぞれｋ番目のＰＥのＳＭに配置され、他のＰＥから参照・変更される可能性があるものとする。

【００３６】

たとえば変数ｉｎの格納場所は、実際にはＰＥ＃１のＳＭ上の一点であり、同じ場所にＰＥ＃０は０ｘ３０００、ＰＥ＃１は０ｘ２０００のアドレスを割り当てているので、いずれのＰＥからもその値を参照・変更することができる。これにより、共有メモリを介したＰＥ間でのデータの授受が可能となっている。

【００３７】

（実施の形態）

次に図１２は、本発明の実施の形態にかかるロードモジュール生成装置のハードウエア構成の一例を示すブロック図である。

【００３８】

図中、まずＣＰＵ１２０１は装置全体の制御を司る。ＲＯＭ１２０２はブートプログラムなどを記憶している。ＲＡＭ１２０３はＣＰＵ１２０１のワークエリアとして使用される。ＨＤＤ１２０４は、ＣＰＵ１２０１の制御にしたがってＨＤ１２０５に対するデータのリード／ライトを制御する。ＨＤ１２０５は、ＨＤＤ１２０４の制御にしたがって書き込まれたデータを記憶する。

【００３９】

ＦＤＤ１２０６は、ＣＰＵ１２０１の制御にしたがってＦＤ１２０７に対するデータのリード／ライトを制御する。ＦＤ１２０７は、ＦＤＤ１２０６の制御にしたがって書き込まれたデータを記憶したり、記憶しているデータをＦＤＤ１２０６の磁気ヘッドに読み取らせたりする。着脱可能な記録媒体としては、ＦＤ１２０７のほかＣＤ－ＲＯＭ、ＣＤ－Ｒ、ＣＤ－ＲＷ、ＭＯ、ＤＶＤ（Ｄｉｇｉｔａｌ　Ｖｅｒｓａｔｉｌｅ　Ｄｉｓｋ）、メモリカードなどが考えられる。

【００４０】

ディスプレイ１２０８は、たとえばＣＲＴ、ＴＦＴ液晶ディスプレイ、プラズマディスプレイなどであって、カーソルやウィンドウをはじめ、文書、画像などの各種データを表示する。ネットワークＩ／Ｆ１２０９は、イーサネット（Ｒ）

ケーブル１２１０を通じてＬＡＮに接続されるとともに、ＬＡＮと装置内部とのデータの送受信を司る。

【００４１】

キーボード１２１１は、文字、数値、各種指示などの入力のためのキーを備え、装置内部へのデータの入力をおこなう。タッチパネル式の入力パッドやテンキーなどであってもよい。マウス１２１２は、カーソルの移動や範囲選択などをおこなう。ポインティングデバイスとして同様の機能を備えるものであれば、トラックボール、ジョイスティック、十字キー、ジョグダイヤルなどであってもよい。なお、上記各部はバスまたはケーブル１２００により接続されている。

【００４２】

次に、図１３は本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。同図に示す各機能部は、具体的には図１２に示したＨＤ１２０５、ＦＤ１２０７などに格納されたプログラム、具体的にはコンパイラ、アセンブラおよびリンカの三つのプログラムを、ＣＰＵ１２０１がＲＡＭ１２０３に読み出して実行することにより実現される。

【００４３】

図中、１３００～１３０２はコンパイラにより実現され、プログラムのソース記述をアセンブリ記述に変換する機能部である。その詳細な機能は従来技術によるコンパイラと同一である。

【００４４】

すなわち第１解析部１３００は、図４に示したステップＳ４０１の処理をおこなう機能部であり、指定されたプログラムのソース記述を読み込んで字句解析および構文解析をおこなうとともに、当該プログラムをコンパイラの内部表現へと変換する。

【００４５】

次に、命令列生成部１３０１は図４に示したステップＳ４０２の処理をおこなう機能部であり、上記内部表現にもとづいてプログラムの動作を実現する命令列を生成するとともに、当該命令列をコンパイラの内部情報に付加する。

【００４６】

次に、アセンブリ記述出力部１３０２は図４に示したステップＳ４０３の処理をおこなう機能部であり、コンパイラの内部表現および付加されている命令列にもとづいて、上記プログラムのアセンブリ記述を出力する。

【００４７】

また、図１３中１３０３～１３０５はアセンブラにより実現され、コンパイラから出力されたアセンブリ記述をさらにオブジェクトに変換する機能部である。その詳細な機能は従来技術によるアセンブラと同一である。

【００４８】

すなわち第２解析部１３０３は、図４に示したステップＳ４０４の処理をおこなう機能部であり、コンパイラのアセンブリ記述出力部１３０２から出力されたアセンブリ記述を読み込んで、字句解析をおこなうとともにアセンブラの内部表現へと変換する。

【００４９】

次に、バイナリ・コード生成部１３０４は図４に示したステップＳ４０５の処理をおこなう機能部であり、アセンブラの内部表現にもとづいてバイナリ・コード（命令コードを含む）を生成し、当該コードをアセンブラの内部情報に付加する。

【００５０】

次に、オブジェクト出力部１３０５は図４に示したステップＳ４０６の処理をおこなう機能部であり、アセンブラの内部表現および付加されているバイナリ・コードにもとづいて、上記プログラムのオブジェクトを出力する。

【００５１】

また、図１３中１３０６～１３１１はリンカにより実現され、アセンブラから出力されたオブジェクトを結合して実行可能なロードモジュールを出力する機能部である。１３０７以下の各部の機能については、後述するフローチャートで説明するが、１３０６についてのみ先に説明する。

【００５２】

図１４は、メモリ空間定義情報記憶部１３０６に保持されるメモリ空間定義情報の内容を模式的に示す説明図である。メモリ空間定義情報とは、上述の「ｔｅ

ｘｔ　ａｒｅａ」「ｄａｔａ　ａｒｅａ」などの各メモリ領域を、メモリ空間内のどのアドレスに配置するかをＰＥごとに定義したものである。

【００５３】

図中、たとえばＰＥ＃０のメモリ空間上で０ｘ０００００から０ｘ０ｆｆｆまでのアドレスが指し示す場所は、物理的にはＰＥ＃０のＬＭ上に存在し、ここにＰＥ＃０用のプログラムの「ｔｅｘｔ　ａｒｅａ」が配置される。同様に、ＰＥ＃１のメモリ空間上で０ｘ０００００から０ｘ０ｆｆｆまでのアドレスが指し示す場所は、物理的にはＰＥ＃１のＬＭ上に存在し、ここにＰＥ＃１用のプログラムの「ｔｅｘｔ　ａｒｅａ」が配置される。

【００５４】

また、ＰＥ＃０のメモリ空間上で０ｘ３０００から０ｘ３ｆｆｆまでのアドレスが指し示す場所と、ＰＥ＃１のメモリ空間上で０ｘ２０００から０ｘ２ｆｆｆまでのアドレスが指し示す場所とは同一であり、物理的にはＰＥ＃１のＳＭ上に存在する。そして、ここに「ｓｈａｒｅｄ　ｄａｔａ　ａｒｅａ　＃１」、すなわちＰＥ＃０からも参照・変更可能なＰＥ＃１の共有データが配置される。

【００５５】

次に、図１５は本発明の実施の形態にかかるロードモジュール生成装置における、図９および図１０に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。もっとも、コンパイラおよびアセンブラによる処理は従来技術と同様であるので、同図にはリンカによる処理の手順のみを示している。

【００５６】

まず、リンカにより実現されるオブジェクト読み込み部１３０７が、アセンブラのオブジェクト出力部１３０５から出力されたオブジェクトのうち、ｋ番目（０≦ｋ≦ｎ）のＰＥ用のものをリンカの内部表現として読み込む（ステップＳ１５０１）。

【００５７】

次にメモリ空間構築部１３０８が、リンカの内部表現において、ｋ番目のＰＥの各メモリ領域（上述の「ｔｅｘｔ　ａｒｅａ」や「ｄａｔａ　ａｒｅａ」など

）を形成し、リンカの内部表現として付加する（ステップＳ１５０２）。

【００５８】

次にメモリ空間内アドレス解決部１３０９が、リンカの内部表現において、ｋ番目のＰＥのメモリ空間内の各メモリ領域のアドレス解決をおこなう（ステップＳ１５０３）。そして、ここまでの処理をＰＥ＃０（ｋ＝０）からＰＥ＃ｎ（ｋ＝ｎ）のすべてについておこなう。

【００５９】

次にメモリ空間間アドレス解決部１３１０は、上記の処理で得られた各ＰＥのメモリ空間イメージと、図１４に示したメモリ空間定義情報とを参照して、ステップＳ１５０３による各メモリ空間内でのアドレス解決では未解決のまま残ったシンボルについて、メモリ空間をまたがったアドレス解決をおこなう（ステップＳ１５０４）。

【００６０】

メモリ空間をまたがったアドレス解決とは、具体的にはたとえば図９に示したＰＥ＃０用のプログラムにおいて、ＰＥ＃１用のプログラムで宣言されているためにアドレスが解決できない変数ｉｎおよびｏｕｔについて、当該変数のＰＥ＃１のメモリ空間上でのアドレスから、ＰＥ＃０のメモリ空間上でのアドレスを算出する処理である。

【００６１】

あるシンボルのあるＰＥにおけるアドレスを、他ＰＥにおけるアドレスから算出するための計算式は下記の通りである。

ｓｙｍｂｏｌ　ａｄｄｒｅｓｓ＝ｓｅｌｆ　ｂａｓｅ　ａｄｄｒｅｓｓ＋ｏｆｆｓｅｔ

ただしｏｆｆｓｅｔ＝ｏｔｈｅｒ’ｓ　ＰＥ　ｓｙｍｂｏｌ　ａｄｄｒｅｓｓ－ｏｔｈｅｒ’ｓ　ＰＥ　ｂａｓｅ　ａｄｄｒｅｓｓ

【００６２】

たとえば変数ｏｕｔのＰＥ＃０におけるアドレスは、ｏｆｆｓｅｔ＝４（＝０ｘ２００４－０ｘ２０００）であることから、ｓｙｍｂｏｌ　ａｄｄｒｅｓｓ＝０ｘ３００４（＝０ｘ３０００＋０ｘ０００４）となる。すなわち図１４のメモ

リ空間定義情報から、少なくともＰＥ＃０のメモリ空間上でＰＥ＃１との共有データが配置される開始アドレスは分かるので、変数ｏｕｔのオフセットを当該開始アドレスに足し合わせることで、ＰＥ＃０にとっての当該変数のアドレスを割り出している。

【００６３】

メモリ空間内アドレス解決部１３０９およびメモリ空間間アドレス解決部１３１０によるアドレス解決の後は、未解決シンボルは存在しないはずなので、次にロードモジュール出力部１３１１がリンカの内部表現にもとづいて、ｋ番目のＰＥ用のプログラムのロードモジュールを出力する（ステップＳ１５０５）。そして、ＰＥ＃０（ｋ＝０）からＰＥ＃ｎ（ｋ＝ｎ）のすべてについて同様にロードモジュールを出力した時点で、全ＰＥについてソースからロードモジュールまでのプログラムの変換処理が終了する。

【００６４】

なお、図６に示したメモリ空間定義例のもとでは、物理的に他ＰＥのメモリに存在する変数を参照・変更するプログラムはＰＥ＃０用のものに限定される（ＰＥ＃１～＃ｎの各メモリ空間には他ＰＥのメモリが割り当てられていないため、これらのＰＥについて、他ＰＥの変数を参照・変更するようなプログラムは作成することができない）。したがって、ＰＥ＃１～＃ｎ用のプログラムのリンク時には、ステップＳ１５０３の後未解決シンボルは残らないはずなので、ステップＳ１５０４によるアドレス解決は不要である。

【００６５】

もっとも、図６のようにＰＥ＃０のみが他のＰＥのメモリを読み書きできるようにする必然性はなく（図６のような定義は一例に過ぎない）、ＰＥ＃０以外のＰＥからも他ＰＥのメモリを参照・変更できるようにしてもよい。

【００６６】

その場合ＰＥ＃０以外でも、ステップＳ１５０３の処理だけでは解決できないシンボルが発生しうるため、たとえば図１６に示すように、ＰＥ＃１～＃ｎについてもステップＳ１５０４によるアドレス解決をおこなう。ＰＥ＃０について行ったのと同様の処理を他ＰＥについてもおこなうというのみで、図１５の手順と

処理の内容が異なるわけではない。逆に図１５の手順は、図１６においてＰＥ＃０以外のＰＥにつきステップＳ１５０４を省略できる特別の場合であるとみることもできる。

【００６７】

以上説明したように、本発明によれば分散共有メモリ型マルチプロセッサ方式に対応した、ＭＰＭＤプログラミングにもとづくプログラムのロードモジュールを生成することが可能である。逆にいえば本発明により、分散共有メモリ型マルチプロセッサ方式を前提とするプログラムを、ＭＰＭＤプログラミングにより作成することができるので、各ＰＥにはそこで実行される部分プログラムのみを配置することができ、チップに搭載するメモリの容量を少なくすることができる。

【００６８】

なお、本実施の形態におけるロードモジュール生成方法は、あらかじめ用意されたプログラム（コンパイラ、アセンブラおよびリンカ）がパーソナルコンピュータ、ワークステーションなどの各種のコンピュータ上で実行されることにより実現されるが、このプログラムはＨＤ、ＦＤ、ＣＤ－ＲＯＭ、ＭＯ、ＤＶＤなどのコンピュータで読み取り可能な各種の記録媒体に記録され、当該記録媒体によって配布することができるほか、インターネットなどのネットワークを介して配布することも可能である。

【００６９】

（付記１）複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第１のメモリ空間構築工程と、

前記第１のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第１のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第１のメモリ空間内アドレス解決工程と、

第２のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第２のメモリ空間構築工程と、

前記第２のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづ

いて、前記第２のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第２のメモリ空間内アドレス解決工程と、

前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

を含んだことを特徴とするロードモジュール生成方法。

【００７０】

（付記２）前記メモリ空間間アドレス解決工程では、前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第２のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記１に記載のロードモジュール生成方法。

【００７１】

（付記３）前記第１のメモリ空間構築工程では、前記第１のプロセッサからのみ参照可能なメモリ領域と、前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記１または付記２に記載のロードモジュール生成方法。

【００７２】

（付記４）前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域は、前記第２のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記３に記載のロードモジュール生成方法。

【００７３】

（付記５）複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第１のメモリ空間構築工程と、

前記第１のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第１のプロセッサにより実行されるプログラム中の各シンボルのアド

レスを算出する第１のメモリ空間内アドレス解決工程と、

第２のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第２のメモリ空間構築工程と、

前記第２のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第２のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第２のメモリ空間内アドレス解決工程と、

前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【００７４】

（付記６）前記メモリ空間間アドレス解決工程では、前記第１のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第２のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記５に記載のロードモジュール生成プログラム。

【００７５】

（付記７）前記第１のメモリ空間構築工程では、前記第１のプロセッサからのみ参照可能なメモリ領域と、前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記５または付記６に記載のロードモジュール生成プログラム。

【００７６】

（付記８）前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域は、前記第２のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記７に記載のロードモジュール生成プログラム。

【００７７】

（付記９）複数のプロセッサを搭載する計算機システムにより実行されるプログ

ラムのロードモジュールを生成するロードモジュール生成装置において、

第１のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第１のメモリ空間構築手段と、

前記第１のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第１のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第１のメモリ空間内アドレス解決手段と、

第２のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第２のメモリ空間構築手段と、

前記第２のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第２のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第２のメモリ空間内アドレス解決手段と、

前記第１のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決手段と、

を備えたことを特徴とするロードモジュール生成装置。

【００７８】

（付記１０）前記メモリ空間間アドレス解決手段は、前記第１のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第２のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスの、前記第２のメモリ空間構築手段により構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記９に記載のロードモジュール生成装置。

【００７９】

（付記１１）前記第１のメモリ空間構築手段は、前記第１のプロセッサからのみ参照可能なメモリ領域と、前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記９または付記１０に記載のロードモジュール生成装置。

【００８０】

（付記１２）前記第１のプロセッサおよび前記第２のプロセッサから参照可能なメモリ領域は、前記第２のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記１１に記載のロードモジュール生成装置。

【００８１】

【発明の効果】

以上説明したように本発明によれば、ＭＰＭＤプログラミングにもとづいて作成された各プロセッサ用のソースプログラムから、分散共有メモリ型マルチプロセッサ方式を採用する計算機でも実行可能なそのロードモジュールを生成することができるので（逆にいえば、分散共有メモリ型マルチプロセッサ方式の計算機についても、ＭＰＭＤプログラミングによってプログラムを作成することができるので）、これによって、分散共有メモリ型マルチプロセッサ方式を採用する計算機システムにおいても、より少ないメモリ容量で動作するプログラムのロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置が得られるという効果を奏する。

【図面の簡単な説明】

【図１】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。

【図２】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、メモリ空間の定義例を模式的に示す説明図である。

【図３】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、ＳＰＭＤプログラミングにもとづくプログラムの一例を示す説明図である。

【図４】

図３に示したプログラムのロードモジュールの生成手順を示すフローチャートである。

【図５】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。

【図６】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、メモリ空間の定義例を模式的に示す説明図である。

【図７】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、ＭＰＭＤプログラミングにもとづくプログラムの一例（ＰＥ＃０用）を示す説明図である。

【図８】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、ＭＰＭＤプログラミングにもとづくプログラムの一例（ＰＥ＃１～＃ｎ用）を示す説明図である。

【図９】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、ＭＰＭＤプログラミングにもとづくプログラムの一例（ＰＥ＃０用）を示す説明図である。

【図１０】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、ＭＰＭＤプログラミングにもとづくプログラムの一例（ＰＥ＃１用）を示す説明図である。

【図１１】

図９および図１０に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図である。

【図１２】

本発明の実施の形態にかかるロードモジュール生成装置のハードウエア構成の一例を示すブロック図である。

【図１３】

本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示す

ブロック図である。

【図１４】

メモリ空間定義情報記憶部１３０６に保持されるメモリ空間定義情報の内容を模式的に示す説明図である。

【図１５】

本発明の実施の形態にかかるロードモジュール生成装置における、図９および図１０に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。

【図１６】

本発明の実施の形態にかかるロードモジュール生成装置における、図９および図１０に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。

【符号の説明】

１２００　バスまたはケーブル
１２０１　ＣＰＵ
１２０２　ＲＯＭ
１２０３　ＲＡＭ
１２０４　ＨＤＤ
１２０５　ＨＤ
１２０６　ＦＤＤ
１２０７　ＦＤ
１２０８　ディスプレイ
１２０９　ネットワークＩ／Ｆ
１２１０　イーサネット（Ｒ）ケーブル
１２１１　キーボード
１２１２　マウス
１３００　第１解析部
１３０１　命令列生成部
１３０２　アセンブリ記述出力部

１３０３　第２解析部

１３０４　バイナリ・コード生成部

１３０５　オブジェクト出力部

１３０６　メモリ空間定義情報記憶部

１３０７　オブジェクト読み込み部

１３０８　メモリ空間構築部

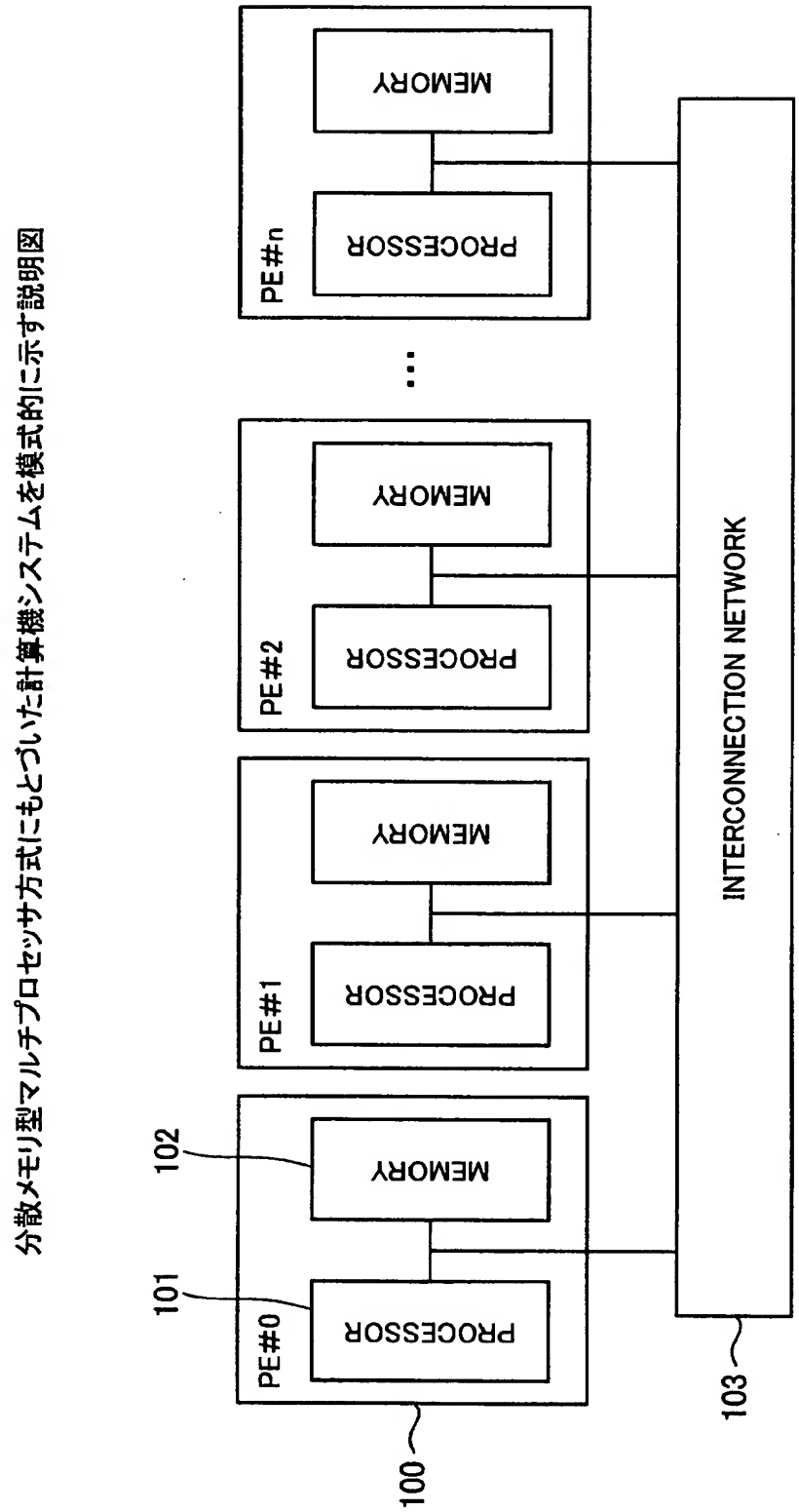
１３０９　メモリ空間内アドレス解決部

１３１０　メモリ空間間アドレス解決部

１３１１　ロードモジュール出力部

【書類名】　　図面

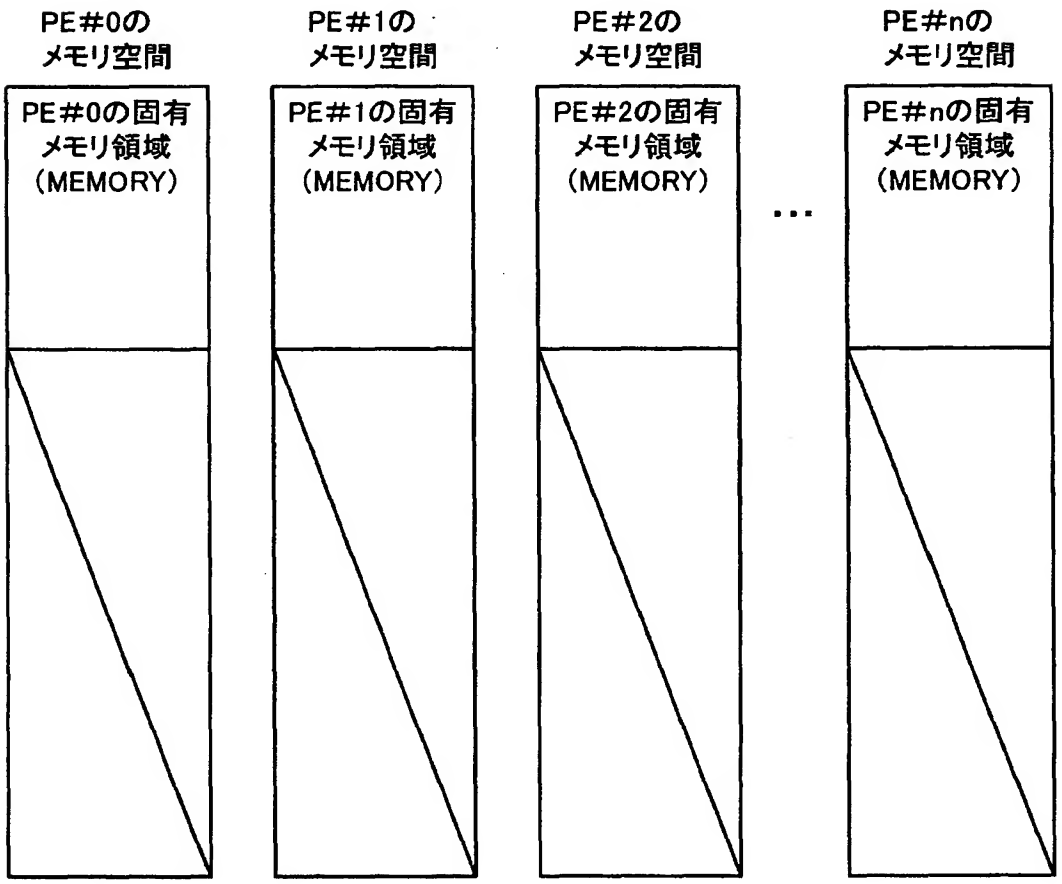
【図１】



出証特２００２－３１０３５６５

【図２】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、
メモリ空間の定義例を模式的に示す説明図



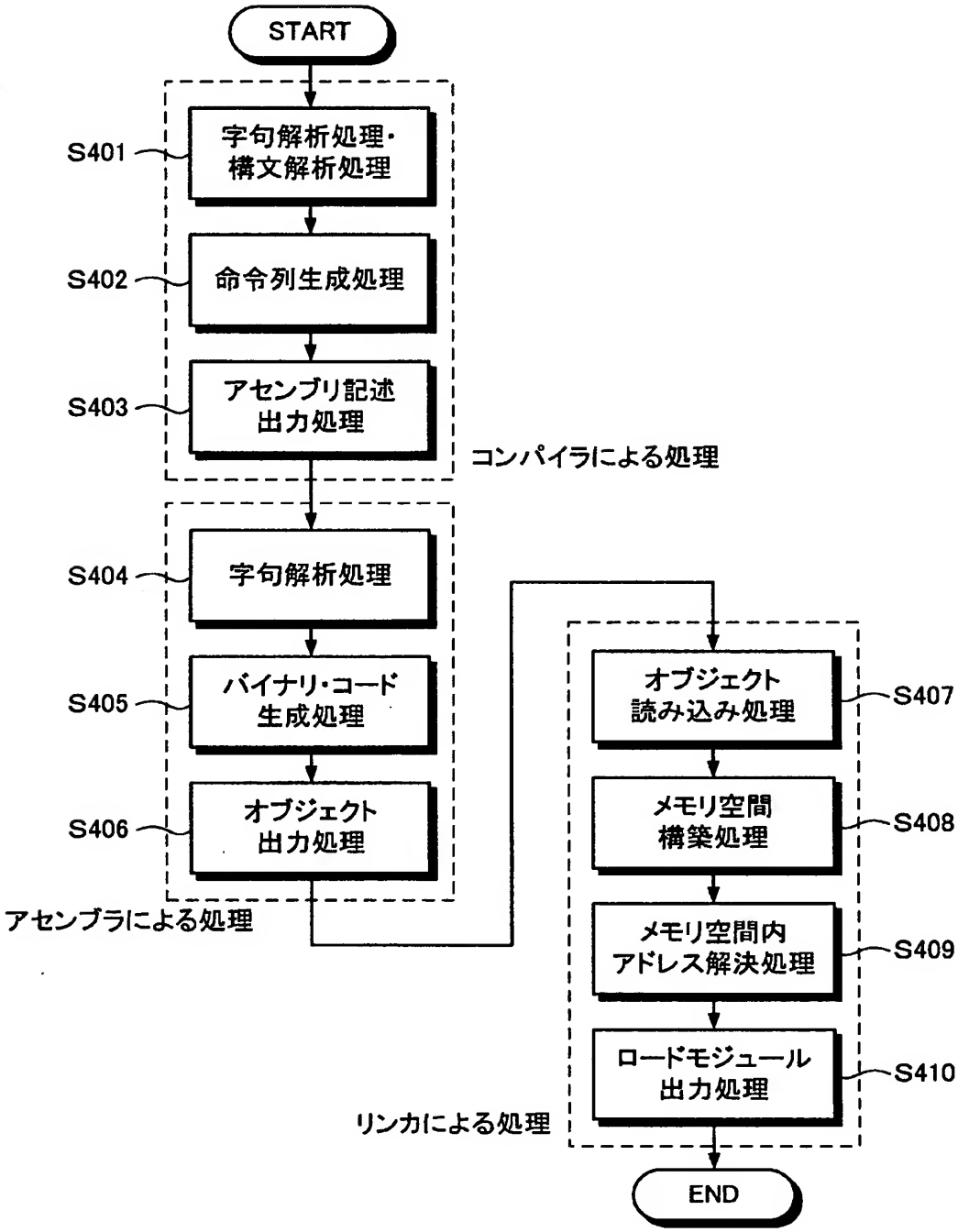
【図３】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
SPMDプログラミングにもとづくプログラムの一例を示す説明図

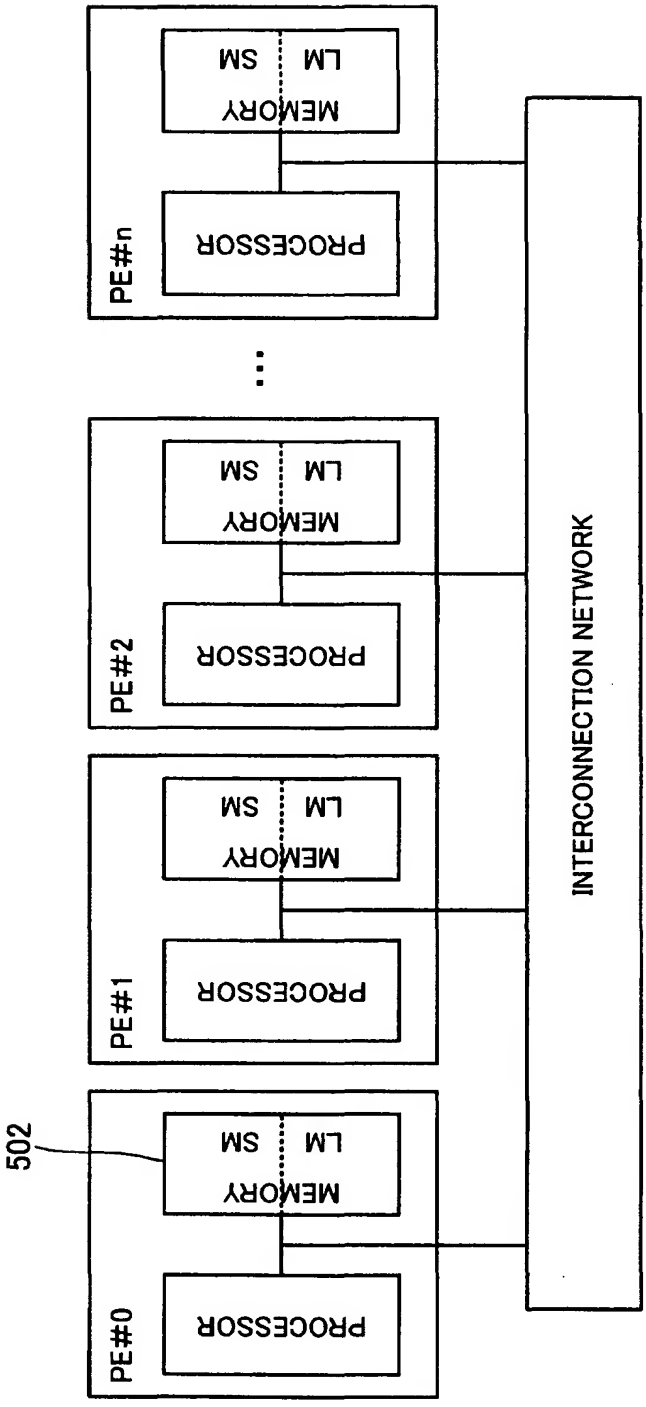
```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank;/* カレントプロセスのランク */
    int source;/* 送信プロセスのランク */
    int dest;/* 受信プロセスのランク */
    int tag=0;/* メッセージのタグ */
    char message[100];/* メッセージの保管場所 */
    MPI_Status status;/* 受信の戻りステータス */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);


    if (my_rank != 0) {
      /* メッセージの生成 */
      sprintf(message, "Greetings from process %d\n", my_rank);
      dest=0;
      /*'\0'も送信するので strlen+1 を使う */
      MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
      MPI_COMM_WORLD);
    } else {
      source=1;
      MPI_Recv(message, sizeof(message), MPI_CHAR, source, tag,
      MPI_COMM_WORLD, &status);
      printf("%s\n", message);
    }

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

【図４】

図3に示したプログラムのロードモジュールの生成手順を示すフローチャート



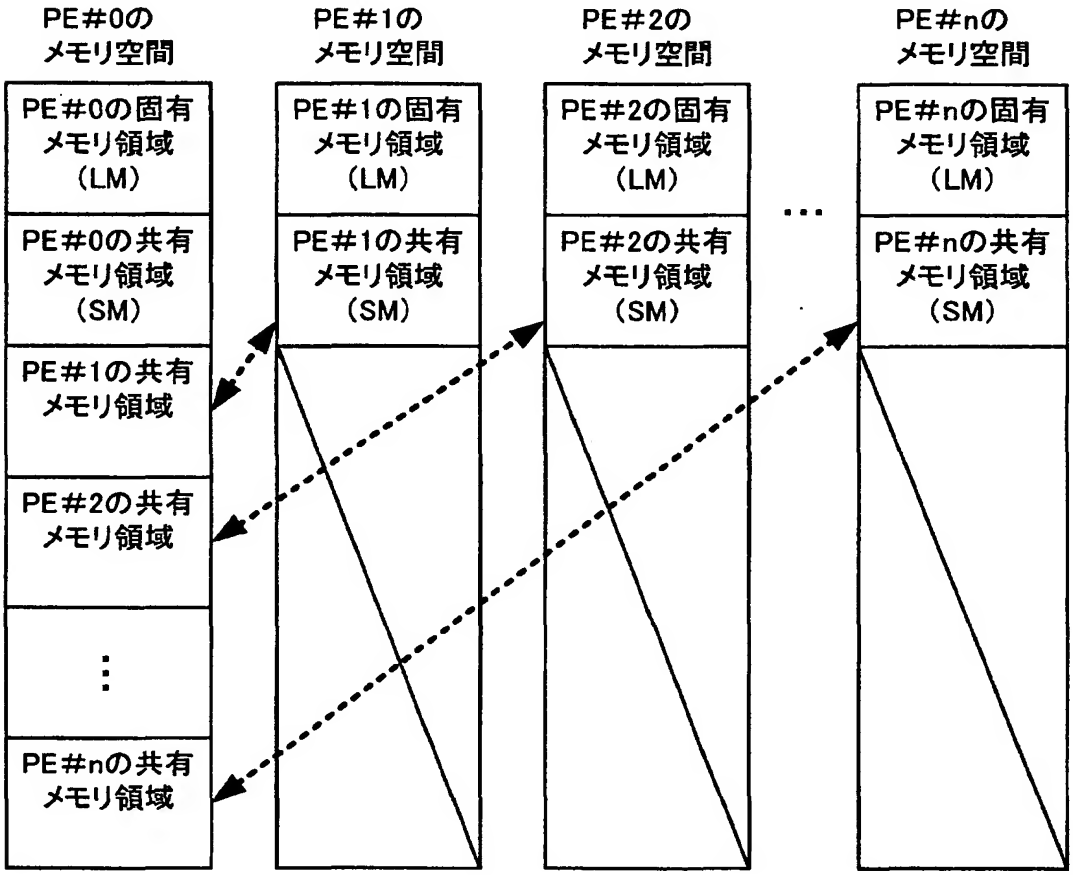
【図５】



分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図

【図６】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、
メモリ空間の定義例を模式的に示す説明図



【図７】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#0用)を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
      int my_rank;/* カレントプロセスのランク */
      int source;/* 送信プロセスのランク */
      int tag=0;/* メッセージのタグ */
      char message[100];/* メッセージの保管場所 */
      MPI_Status status;/* 受信の戻りステータス */

      /* MPI のスタートアップ */
      MPI_Init(&argc, argv);

      /* カレントプロセスのランクを求める */
      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

      source=1;
      MPI_Recv(message, sizeof(message), MPI_CHAR, source, tag,
      MPI_COMM_WORLD, &status);
      printf("%s\n", message);

      /* MPI のシャットダウン */
      MPI_Finalize();
      return 0;
}
```

【図８】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#1～#n用)を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank;/* カレントプロセスのランク */
    int dest;/* 受信プロセスのランク */
    int tag=0;/* メッセージのタグ */
    char message[100];/* メッセージの保管場所 */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    /* メッセージの生成 */
    sprintf(message, "Greetings from process %d¥n", my_rank);
    dest=0;
    /*'¥0'も送信するので strlen+1 を使う */
    MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
    MPI_COMM_WORLD);

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

【図９】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#0用)を示す説明図

```
int input;
int output;
extern int in;
extern int out;


void
Th0(void)
{
    MOVE(&in, &input, sizeof(in));          /* Th0-1 */
    START(1."Th1");                         /* Th0-2 */
    MOVE(&output, &out, sizeof(output)); /* Th0-3 */
}
```

【図１０】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#1用)を示す説明図

```
int in;
int out;

void
Th1(void)
{
    extern void f1(int *, int *);

    f1(&in, &out);                          /* Th1-1 */
}
```

【図１１】

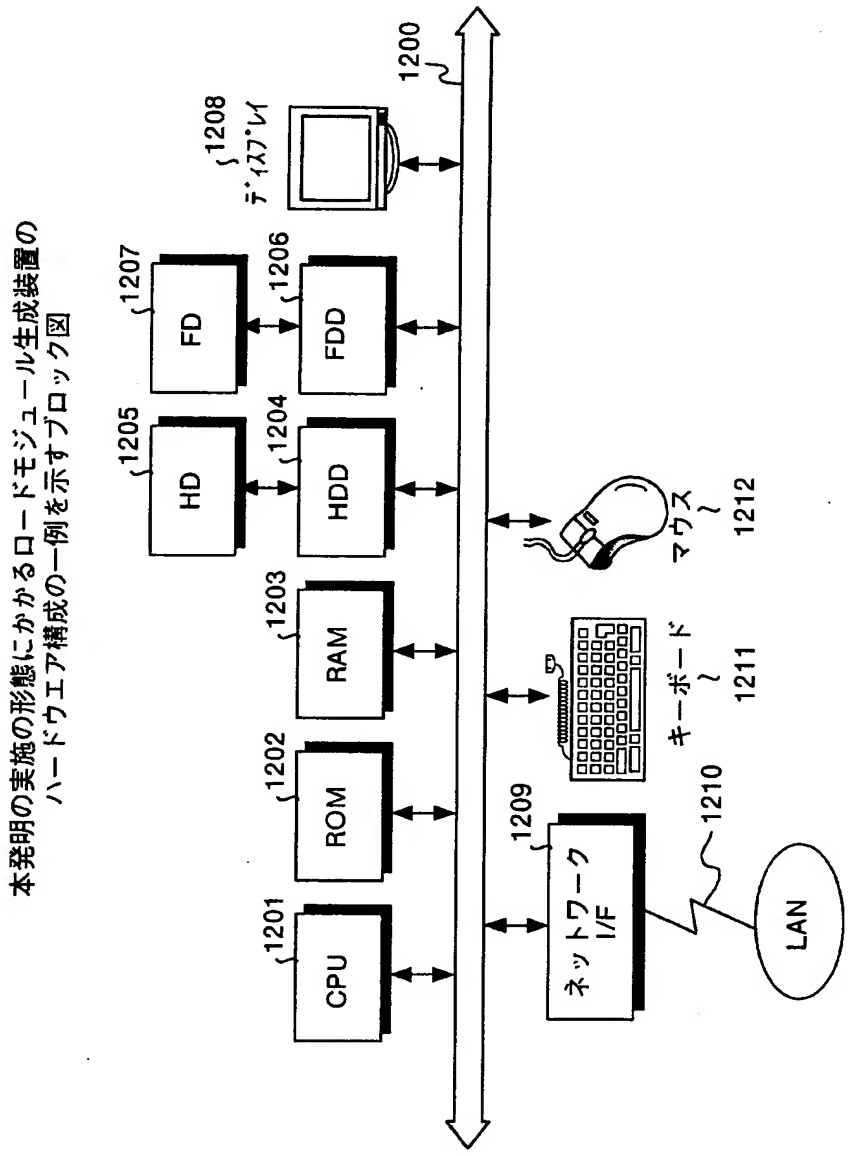
図9および図10に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図

	PE#0のメモリ空間		PE#1のメモリ空間	
	アドレス	内容	アドレス	内容

text area	0x0000	void
		Th0(void)
		{
		MOVE(0x3000, 0x1000, sizeof(in));
		START(1, "Th1");
		MOVE(0x1004, 0x3004, sizeof(output));
		}
			0x0000	void
				Th1(void)
				{
				f1(0x2000, 0x2004));
				}
				
				void
				f1(int *in, int, *out)
				{
				:
				:
				:
				}

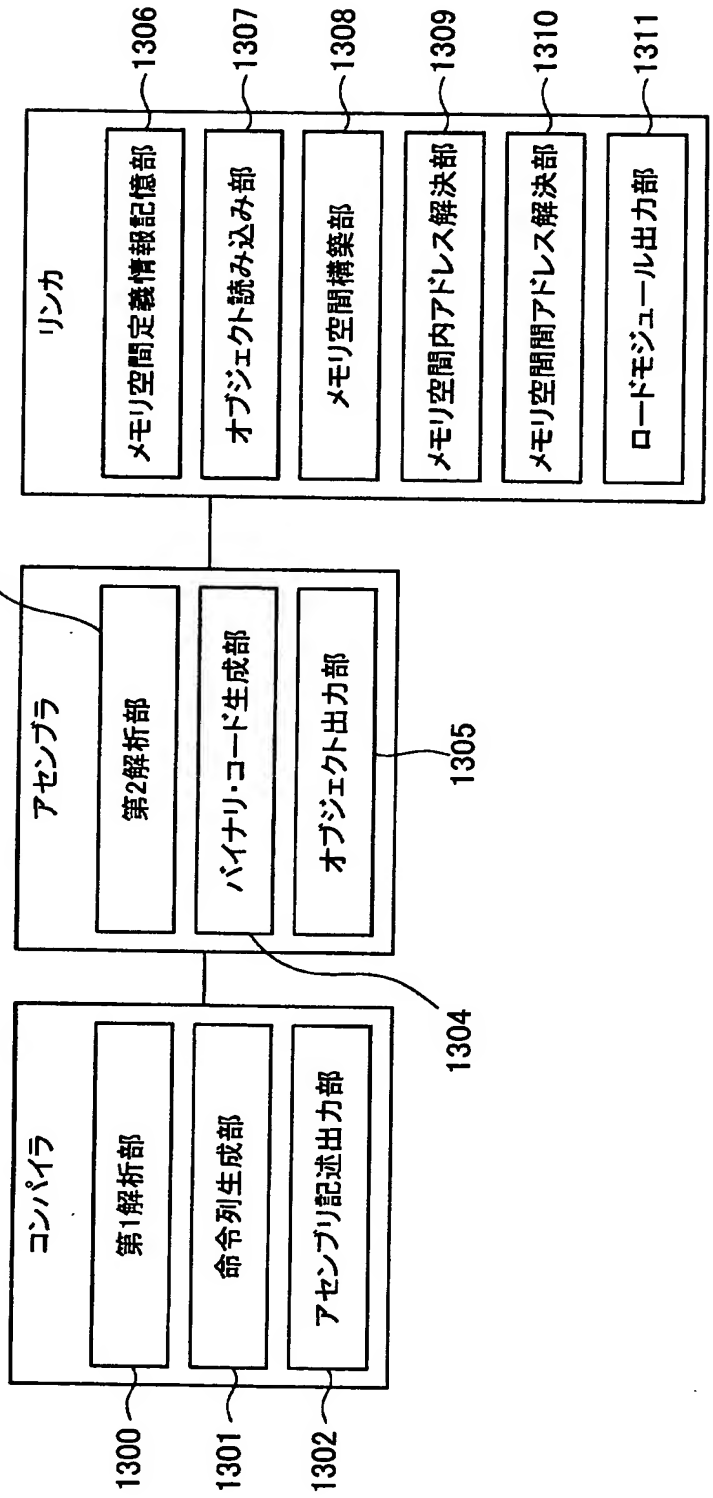
data area	0x1000	int input;	0x1000	
	0x1004	int output;		
shared data area #0	0x2000			
shared data area #1	0x3000	int in;	0x2000	int in;
	0x3004	int out;	0x2004	int out;

【図１２】


本発明の実施の形態にかかるロードモジュール生成装置の
ハードウエア構成の一例を示すブロック図
1200
1208
ディスプレイ
1207
FD
1206
FDD
1205
HD
1204
HDD
1203
RAM
1202
ROM
1201
CPU
1209
ネットワーク
I/F
1210
LAN
キーボード
1211
マウス
1212

【図１３】

本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示すブロック図



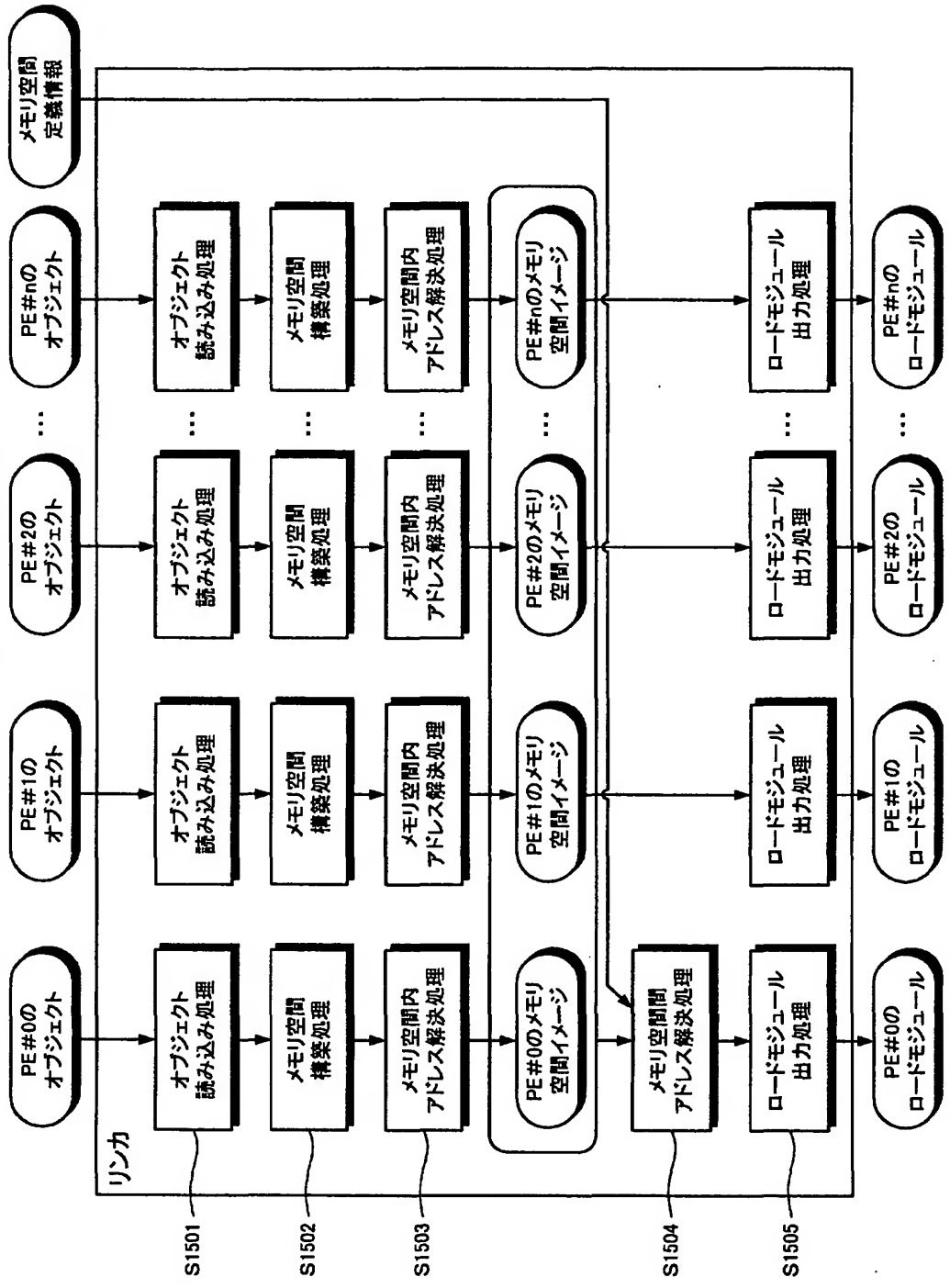
【図１４】

メモリ空間定義情報記憶部1306に保持されるメモリ空間定義情報の内容を模式的に示す説明図

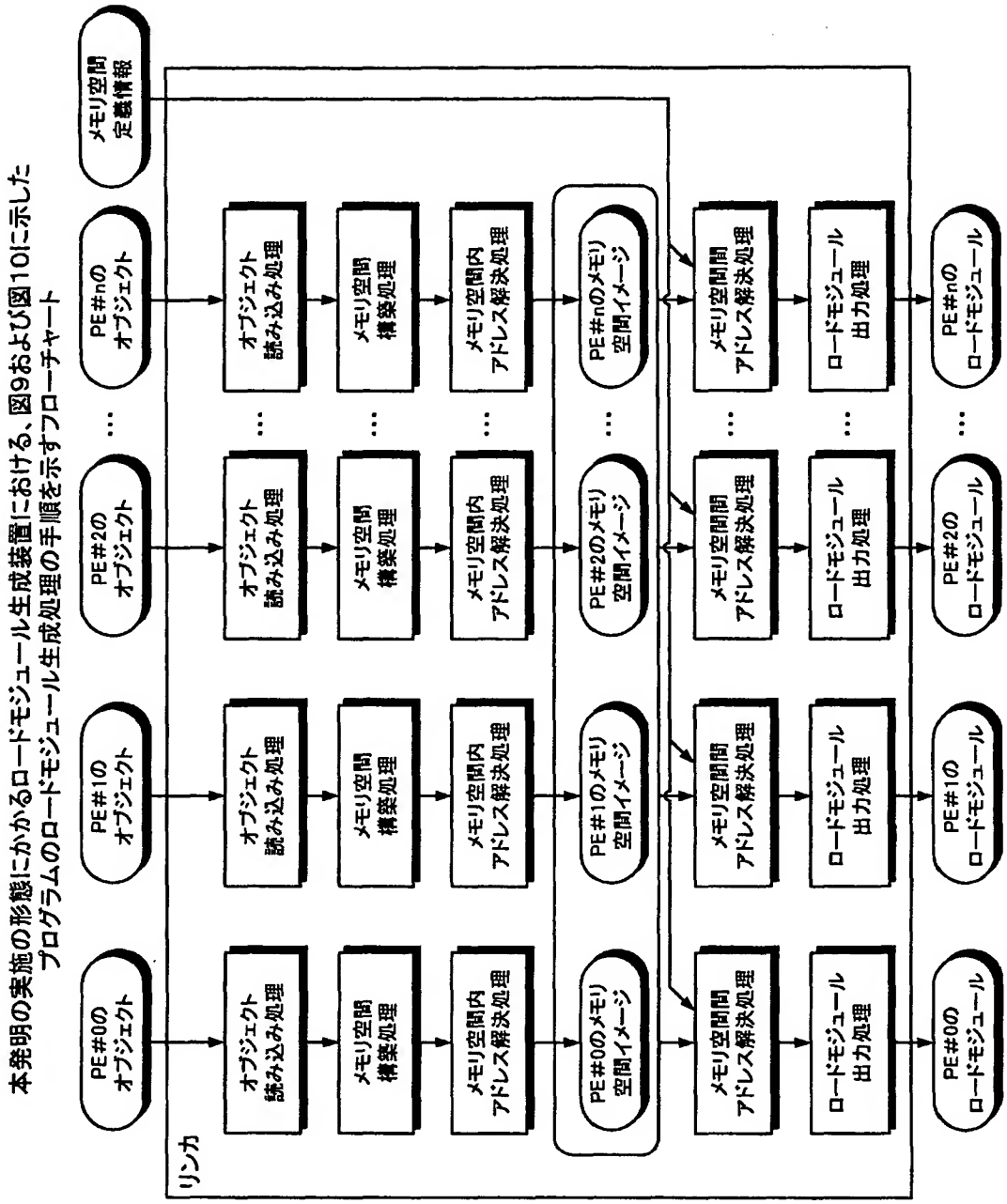
PE識別	領域名	開始アドレス	終了アドレス
PE#0	text area	0x0000	0x0fff
	data area	0x1000	0x1fff
	shared data area #0	0x2000	0x2fff
	shared data area #1	0x3000	0x3fff
PE#1	text area	0x0000	0x0fff
	data area	0x1000	0x1fff
	shared data area #1	0x2000	0x2fff

【図１５】

本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャート



【図１６】


本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示した
プログラムのロードモジュール生成処理の手順を示すフローチャート
メモリ空間
定義情報
PE#0の
オブジェクト
PE#1の
オブジェクト
PE#2の
オブジェクト
PE#nの
オブジェクト
オブジェクト
読み込み処理
メモリ空間
構築処理
メモリ空間内
アドレス解決処理
PE#0のメモリ
空間イメージ
PE#1のメモリ
空間イメージ
PE#2のメモリ
空間イメージ
PE#nのメモリ
空間イメージ
メモリ空間間
アドレス解決処理
ロードモジュール
出力処理
PE#0の
ロードモジュール
PE#1の
ロードモジュール
PE#2の
ロードモジュール
PE#nの
ロードモジュール
リンカ

【書類名】　　要約書

【要約】

【課題】　分散共有メモリ型マルチプロセッサ方式を採用する計算機システムにおいて、ＳＰＭＤでなくＭＰＭＤプログラミングにもとづくプログラムを動作させることでメモリの有効利用をはかるべく、現実に実行可能な（すなわち、プログラム中の全シンボルのアドレスが解決された）当該プログラムのロードモジュールを生成するための言語処理系を提供すること。

【解決手段】分散共有メモリ型マルチプロセッサ方式を前提とするＭＰＭＤプログラミングでは、たとえばＰＥ＃０用のプログラムがＰＥ＃１の共有メモリ上のデータを参照・変更することがあるが、当該データのアドレス（メモリ空間上での位置）はＰＥごとに異なっている。そこでメモリ空間間アドレス解決部１３１０により、ＰＥ＃１にとっての上記データのアドレスから、そのＰＥ＃０にとってのアドレスを所定の計算式にもとづいて算出する。

【選択図】　　図１３

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　富士通株式会社